

i.MX8DXL and i.MX95 SHE(Secure Hardware Extension) API Rev 0.1

NXP Copyright

Generated by Doxygen 1.8.17

| | |
|---|----------|
| 1 SHE(Secure Hardware Extension) API | 1 |
| 2 Revision History | 3 |
| 3 General concepts related to the API | 5 |
| 3.1 Session | 5 |
| 3.2 Service flow | 5 |
| 3.3 Key store | 5 |
| 3.4 Implementation specificities | 5 |
| 4 Module Index | 7 |
| 4.1 Modules | 7 |
| 5 Module Documentation | 9 |
| 5.1 CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB | 9 |
| 5.1.1 Detailed Description | 10 |
| 5.1.2 Data Structure Documentation | 10 |
| 5.1.2.1 struct open_svc_cipher_args_t | 10 |
| 5.1.2.2 struct op_cipher_one_go_args_t | 10 |
| 5.1.3 Function Documentation | 11 |
| 5.1.3.1 she_open_cipher_service() | 11 |
| 5.1.3.2 she_close_cipher_service() | 11 |
| 5.1.3.3 she_cipher_one_go() | 11 |
| 5.2 CMD_EXPORT_RAM_KEY | 14 |
| 5.2.1 Detailed Description | 14 |
| 5.2.2 Data Structure Documentation | 14 |
| 5.2.2.1 struct op_export_plain_key_args_t | 14 |
| 5.2.3 Function Documentation | 14 |
| 5.2.3.1 she_export_plain_key() | 14 |
| 5.3 CMD_GENERATE_MAC | 16 |
| 5.3.1 Detailed Description | 16 |
| 5.3.2 Data Structure Documentation | 16 |
| 5.3.2.1 struct op_generate_mac_t | 16 |
| 5.3.3 Function Documentation | 16 |
| 5.3.3.1 she_generate_mac() | 17 |
| 5.3.3.2 she_generate_fast_mac_mubuff_v2() | 17 |
| 5.4 CMD_VERIFY_MAC | 18 |
| 5.4.1 Detailed Description | 18 |
| 5.4.2 Data Structure Documentation | 18 |
| 5.4.2.1 struct op_verify_mac_t | 18 |
| 5.4.3 Function Documentation | 19 |
| 5.4.3.1 she_verify_mac() | 19 |
| 5.4.3.2 she_verify_fast_mac_mubuff_v2() | 19 |
| 5.5 CMD_GET_ID | 20 |

| | |
|---|----|
| 5.5.1 Detailed Description | 20 |
| 5.5.2 Data Structure Documentation | 20 |
| 5.5.2.1 struct op_get_id_args_t | 20 |
| 5.5.3 Function Documentation | 20 |
| 5.5.3.1 she_get_id() | 20 |
| 5.6 SHE Get Info | 22 |
| 5.6.1 Detailed Description | 22 |
| 5.6.2 Data Structure Documentation | 22 |
| 5.6.2.1 struct op_get_info_args_t | 22 |
| 5.6.3 Function Documentation | 23 |
| 5.6.3.1 she_get_info() | 23 |
| 5.7 SHE Commands | 24 |
| 5.7.1 Detailed Description | 24 |
| 5.8 CMD_GET_STATUS | 25 |
| 5.8.1 Detailed Description | 25 |
| 5.8.2 Data Structure Documentation | 25 |
| 5.8.2.1 struct op_get_status_args_t | 25 |
| 5.8.3 Function Documentation | 25 |
| 5.8.3.1 she_get_status() | 25 |
| 5.9 Session | 27 |
| 5.9.1 Detailed Description | 28 |
| 5.9.2 Data Structure Documentation | 28 |
| 5.9.2.1 struct she_session_hdl_s | 28 |
| 5.9.2.2 struct she_service_hdl_s | 28 |
| 5.9.2.3 struct open_session_args_t | 28 |
| 5.9.3 Typedef Documentation | 29 |
| 5.9.3.1 she_hdl_t | 29 |
| 5.9.4 Function Documentation | 29 |
| 5.9.4.1 she_session_hdl_to_ptr() | 29 |
| 5.9.4.2 delete_she_session() | 29 |
| 5.9.4.3 add_she_session() | 30 |
| 5.9.4.4 she_service_hdl_to_ptr() | 30 |
| 5.9.4.5 delete_she_service() | 30 |
| 5.9.4.6 add_she_service() | 30 |
| 5.9.4.7 she_open_session() | 31 |
| 5.9.4.8 she_close_session() | 31 |
| 5.10 SHE keys | 32 |
| 5.10.1 Detailed Description | 32 |
| 5.11 SHE+ key extension | 33 |
| 5.11.1 Detailed Description | 33 |
| 5.12 Key store | 34 |
| 5.12.1 Detailed Description | 34 |

| | |
|---|----|
| 5.12.2 Data Structure Documentation | 34 |
| 5.12.2.1 struct open_svc_key_store_args_t | 34 |
| 5.12.2.2 struct op_key_store_reprov_en_args_t | 35 |
| 5.12.3 Function Documentation | 35 |
| 5.12.3.1 she_open_key_store_service() | 36 |
| 5.12.3.2 she_close_key_store_service() | 36 |
| 5.13 CMD_LOAD_KEY | 37 |
| 5.13.1 Detailed Description | 37 |
| 5.13.2 Data Structure Documentation | 37 |
| 5.13.2.1 struct op_key_update_args_t | 37 |
| 5.13.2.2 struct op_key_update_ext_args_t | 38 |
| 5.13.3 Function Documentation | 38 |
| 5.13.3.1 she_key_update() | 38 |
| 5.13.3.2 she_key_update_ext() | 39 |
| 5.14 CMD_LOAD_PLAIN_KEY | 40 |
| 5.14.1 Detailed Description | 40 |
| 5.14.2 Data Structure Documentation | 40 |
| 5.14.2.1 struct op_load_plain_key_args_t | 40 |
| 5.14.3 Function Documentation | 40 |
| 5.14.3.1 she_load_plain_key() | 40 |
| 5.15 CMD_INIT_RNG | 42 |
| 5.15.1 Detailed Description | 42 |
| 5.15.2 Function Documentation | 42 |
| 5.15.2.1 she_open_rng_service() | 42 |
| 5.15.2.2 she_close_rng_service() | 42 |
| 5.16 CMD_RND | 44 |
| 5.16.1 Detailed Description | 44 |
| 5.16.2 Data Structure Documentation | 44 |
| 5.16.2.1 struct open_svc_rng_args_t | 44 |
| 5.16.2.2 struct op_get_random_args_t | 44 |
| 5.16.3 Function Documentation | 45 |
| 5.16.3.1 she_get_random() | 45 |
| 5.17 CMD_EXTEND_SEED | 46 |
| 5.17.1 Detailed Description | 46 |
| 5.17.2 Data Structure Documentation | 46 |
| 5.17.2.1 struct op_rng_extend_seed_t | 46 |
| 5.17.3 Function Documentation | 46 |
| 5.17.3.1 she_extend_seed() | 46 |
| 5.18 Shared Buffer | 48 |
| 5.18.1 Detailed Description | 48 |
| 5.18.2 Data Structure Documentation | 48 |
| 5.18.2.1 struct op_shared_buf_args_t | 48 |

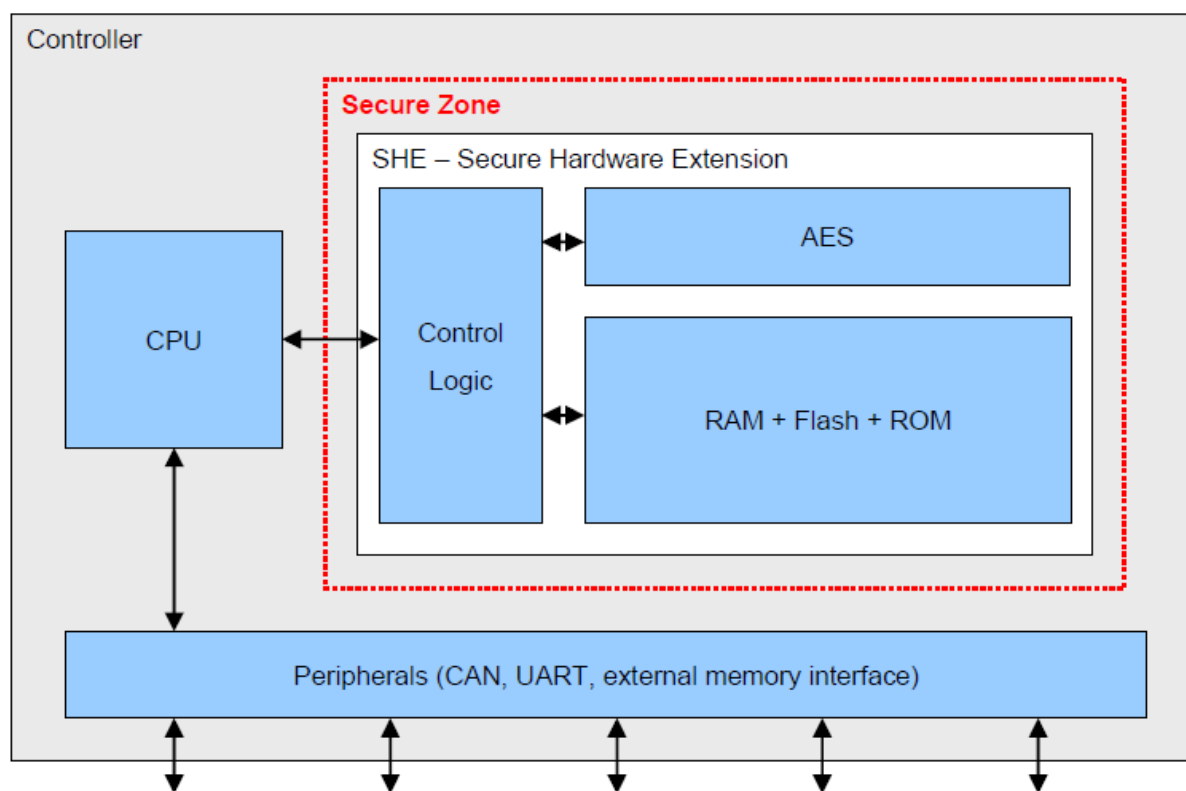
| | |
|---|----|
| 5.19 Error codes | 49 |
| 5.19.1 Detailed Description | 50 |
| 5.19.2 Enumeration Type Documentation | 50 |
| 5.19.2.1 she_err_t | 50 |
| 5.20 Utils | 52 |
| 5.20.1 Detailed Description | 52 |
| 5.20.2 Data Structure Documentation | 52 |
| 5.20.2.1 struct op_open_utils_args_t | 52 |
| 5.20.3 Function Documentation | 52 |
| 5.20.3.1 she_open_utils() | 53 |
| 5.20.3.2 she_close_utils() | 53 |
| 5.21 last rating code | 54 |
| 5.21.1 Detailed Description | 54 |
| 5.21.2 Function Documentation | 54 |
| 5.21.2.1 she_get_last_rating_code() | 54 |
| 5.22 CMD_CANCEL | 55 |
| 5.22.1 Detailed Description | 55 |
| 5.22.2 Function Documentation | 55 |
| 5.22.2.1 she_cmd_cancel() | 55 |
| 5.23 Global Info | 56 |
| 5.23.1 Detailed Description | 57 |
| 5.23.2 Function Documentation | 57 |
| 5.23.2.1 populate_global_info() | 57 |
| 5.23.2.2 show_global_info() | 57 |
| 5.23.2.3 is_global_info_populated() | 57 |
| 5.23.2.4 se_get_soc_id() | 57 |
| 5.23.2.5 se_get_soc_rev() | 57 |
| 5.23.2.6 se_get_chip_lifecycle() | 58 |
| 5.23.2.7 se_get_fips_mode() | 58 |
| 5.23.2.8 se_get_lib_newness_ver() | 58 |
| 5.23.2.9 se_get_lib_major_ver() | 58 |
| 5.23.2.10 se_get_lib_minor_ver() | 58 |
| 5.23.2.11 se_get_nvm_newness_ver() | 58 |
| 5.23.2.12 se_get_nvm_major_ver() | 58 |
| 5.23.2.13 se_get_nvm_minor_ver() | 59 |
| 5.23.2.14 se_get_commit_id() | 59 |
| 5.23.2.15 se_get_lib_version() | 59 |
| 5.23.2.16 se_get_nvm_version() | 59 |
| 5.23.2.17 get_soc_id_str() | 59 |
| 5.23.2.18 get_soc_rev_str() | 60 |
| 5.23.2.19 get_soc_lf_str() | 60 |
| 5.23.2.20 se_get_info() | 60 |

| | |
|---|-----------|
| 5.23.2.21 se_get_soc_info() | 60 |
| Index | 61 |

Chapter 1

SHE(Secure Hardware Extension) API

This document is a software reference description of the API provided by the Secure Enclave Library on i.MX8DXL and i.MX95 platforms for SHE solutions



Chapter 2

Revision History

| Revision | date | description |
|----------|-------------|---------------|
| 0.1 | Jan 13 2024 | Initial Draft |

Chapter 3

General concepts related to the API

3.1 Session

The API must be initialized by a potential requestor by opening a session. The session establishes a route (MU, DomainID...) between the requester and the SHE. When a session is opened, the SHE returns a handle identifying the session to the requester.

3.2 Service flow

For a given category of services which require service handle, the requestor is expected to open a service flow by invoking the appropriate SHE API. The session handle, as well as the control data needed for the service flow, are provided as parameters of the call. Upon reception of the open request, the SHE allocates a context in which the session handle, as well as the provided control parameters are stored and return a handle identifying the service flow.

The context is preserved until the service flow, or the session, are closed by the user and it is used by the SHE to proceed with the sub-sequent operations requested by the user on the service flow.

3.3 Key store

A key store can be created by specifying the CREATE flag in the `she_open_key_store_service` API. Please note that the created key store will be not stored in the NVM till a key is generated or imported specifying the "STRICT OPERATION" flag.

3.4 Implementation specificities

SHE API with common features are supported on i.MX8DXL and i.MX95. The details of supported features per chip will be listed in the platform specificities.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

| | |
|---|----|
| SHE Get Info | 22 |
| SHE Commands | 24 |
| CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB | 9 |
| CMD_EXPORT_RAM_KEY | 14 |
| CMD_GENERATE_MAC | 16 |
| CMD_VERIFY_MAC | 18 |
| CMD_GET_ID | 20 |
| CMD_GET_STATUS | 25 |
| CMD_LOAD_KEY | 37 |
| CMD_LOAD_PLAIN_KEY | 40 |
| CMD_INIT_RNG | 42 |
| CMD_RND | 44 |
| CMD_EXTEND_SEED | 46 |
| last rating code | 54 |
| CMD_CANCEL | 55 |
| Session | 27 |
| SHE keys | 32 |
| SHE+ key extension | 33 |
| Key store | 34 |
| Shared Buffer | 48 |
| Error codes | 49 |
| Utils | 52 |
| Global Info | 56 |

Chapter 5

Module Documentation

5.1 CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB

Data Structures

- struct [open_svc_cipher_args_t](#)
- struct [op_cipher_one_go_args_t](#)

Macros

- #define [SHE_CIPHER_ONE_GO_ALGO_AES_ECB](#) (([she_op_cipher_one_go_algo_t](#))(0x00u))
Indicates it is AES ECB Cipher operation.
- #define [SHE_CIPHER_ONE_GO_ALGO_AES_CBC](#) (([she_op_cipher_one_go_algo_t](#))(0x01u))
Indicates it is AES CBC Cipher operation.
- #define [SHE_CIPHER_ONE_GO_FLAGS_DECRYPT](#) (([she_op_cipher_one_go_flags_t](#))(0u << 0))
Bit indicating the decrypt operation.
- #define [SHE_CIPHER_ONE_GO_FLAGS_ENCRYPT](#) (([she_op_cipher_one_go_flags_t](#))(1u << 0))
Bit indicating the encrypt operation.

Typedefs

- typedef uint8_t [she_op_cipher_one_go_algo_t](#)
Bit field indicating the requested cipher operations.
- typedef uint8_t [she_op_cipher_one_go_flags_t](#)
Bit field indicating the requested encrypt/decrypt operations.

Functions

- [she_err_t she_open_cipher_service](#) ([she_hdl_t](#) session_hdl, [open_svc_cipher_args_t](#) *args)
- [she_err_t she_close_cipher_service](#) ([she_hdl_t](#) cipher_handle)
- [she_err_t she_cipher_one_go](#) ([she_hdl_t](#) cipher_handle, [op_cipher_one_go_args_t](#) *args)

5.1.1 Detailed Description

SHE supports two modes electronic cipher book mode (ECB) for processing single blocks of data and the cipher block chaining mode (CBC) for processing larger amounts of data.

5.1.2 Data Structure Documentation

5.1.2.1 struct open_svc_cipher_args_t

Structure describing the open cipher service members

Data Fields

| | | |
|----------|-------------|--|
| uint32_t | cipher_hdl | handle identifying the cipher service flow |
| uint8_t | flags | bitmap specifying the services properties |
| uint8_t | reserved[3] | reserved bits |

5.1.2.2 struct op_cipher_one_go_args_t

Structure describing the cipher one go operation arguments

Data Fields

| | | |
|-----------|----------------|---|
| uint32_t | key_identifier | identifier of the key to be used for the operation |
| uint8_t * | iv | pointer to the initialization vector (nonce in case of AES CCM) |
| uint16_t | iv_size | length in bytes of the initialization vector. it must be 0 for algorithms not using the initialization vector. It must be 12 for AES in CCM mode (not valid for SHE) |
| uint8_t | svc_flags | bitmap specifying the services properties. |
| uint8_t | flags | bitmap specifying the operation attributes |
| uint8_t | cipher_algo | algorithm to be used for the operation |
| uint8_t * | input | pointer to the input area: <ul style="list-style-type: none"> plaintext for encryption ciphertext for decryption Note: In case of CCM it is the purported ciphertext. |
| uint8_t * | output | pointer to the output area: <ul style="list-style-type: none"> ciphertext for encryption Note: In case of CCM it is the output of the generation-encryption process. plaintext for decryption |
| uint32_t | input_size | length in bytes of the input. <ul style="list-style-type: none"> In case of CBC and ECB, the input size should be multiple of a block cipher size (16 bytes). |
| uint32_t | output_size | length in bytes of the output |

5.1.3 Function Documentation

5.1.3.1 she_open_cipher_service()

```
she_err_t she_open_cipher_service (
    she_hdl_t session_hdl,
    open_svc_cipher_args_t * args )
```

Open a cipher service flow. User can call this function only after having opened a key-store service flow. User must open this service in order to perform cipher operation.

Parameters

| | |
|--------------------|---|
| <i>session_hdl</i> | handle identifying the SHE session. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code.

5.1.3.2 she_close_cipher_service()

```
she_err_t she_close_cipher_service (
    she_hdl_t cipher_handle )
```

Terminate a previously opened cipher service flow

Parameters

| | |
|----------------------|--|
| <i>cipher_handle</i> | handle identifying the Cipher service. |
|----------------------|--|

Returns

error code.

5.1.3.3 she_cipher_one_go()

```
she_err_t she_cipher_one_go (
    she_hdl_t cipher_handle,
    op_cipher_one_go_args_t * args )
```

Perform ciphering operation i.e.

CBC encryption/decryption and ECB encryption/decryption of a given plaintext/ciphertext with the key identified by `key_id`.

User can call this function only after having opened a cipher service flow

Parameters

| | |
|----------------------|---|
| <i>cipher_handle</i> | handle identifying the cipher service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.2 CMD_EXPORT_RAM_KEY

Data Structures

- struct [op_export_plain_key_args_t](#)

Functions

- [she_err_t she_export_plain_key](#) ([she_hdl_t](#) utils_handle, [op_export_plain_key_args_t](#) *args)

5.2.1 Detailed Description

The function exports the RAM_KEY into a format protected by SECRET_KEY. The key can be imported again by using CMD_LOAD_KEY. A RAM_KEY can only be exported if it was written into SHE in plaintext

5.2.2 Data Structure Documentation

5.2.2.1 struct op_export_plain_key_args_t

Structure describing the export RAM key operation arguments

Data Fields

| | | |
|-----------|---------|--|
| uint8_t * | m1 | pointer to the output address for M1 message |
| uint8_t | m1_size | size of M1 message - 128 bits |
| uint8_t * | m2 | pointer to the output address for M2 message |
| uint8_t | m2_size | size of M2 message - 256 bits |
| uint8_t * | m3 | pointer to the output address for M3 message |
| uint8_t | m3_size | size of M3 message - 128 bits |
| uint8_t * | m4 | pointer to the output address for M4 message |
| uint8_t | m4_size | size of M4 message - 256 bits |
| uint8_t * | m5 | pointer to the output address for M5 message |
| uint8_t | m5_size | size of M5 message - 128 bits |

5.2.3 Function Documentation

5.2.3.1 she_export_plain_key()

```
she_err_t she_export_plain_key (
    she_hdl_t utils_handle,
    op_export_plain_key_args_t * args )
```

exports the RAM_KEY into a format protected by SECRET_KEY.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the SHE utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.3 CMD_GENERATE_MAC

Data Structures

- struct [op_generate_mac_t](#)

Macros

- #define [SHE_MAC_SIZE](#) 16u
size of the MAC generated is 128bits.
- #define [SHE_MESSAGE_OFFSET](#) 16u
message is always kept at offset 16 for FAST MAC V2 API.
- #define [SHE_FAST_MAC_FLAGS_GENERATION](#) 0
Flag to generate MAC.

Functions

- [she_err_t she_generate_mac](#) ([she_hdl_t](#) utils_handle, [op_generate_mac_t](#) *args)
- [she_err_t she_generate_fast_mac_mubuff_v2](#) ([she_hdl_t](#) utils_handle, [op_generate_mac_t](#) *args)

5.3.1 Detailed Description

SHE supports two types of functions to generate MAC. 1: for 16 bytes message 2: V2-for 16 or greater than 16 bytes message.

5.3.2 Data Structure Documentation

5.3.2.1 struct op_generate_mac_t

Structure describing the FAST MAC generation operation arguments

Data Fields

| | | |
|-----------|----------------|---|
| uint8_t | flags | optional flag to identify the operation(generate/verify) |
| uint16_t | key_ext | identifier of the key extension to be used for the operation |
| uint16_t | key_id | identifier of the key to be used for the operation |
| uint16_t | message_length | length in bytes of the input message. The message is padded to be a multiple of 128 bits by SHE |
| uint8_t * | message | pointer to the message to be processed |
| uint8_t * | mac | pointer to where the output MAC should be written (128bits should be allocated there) |

5.3.3 Function Documentation

5.3.3.1 she_generate_mac()

```
she_err_t she_generate_mac (
    she_hdl_t utils_handle,
    op_generate_mac_t * args )
```

Generates a MAC of a given message with the help of a key identified by key_id.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.3.3.2 she_generate_fast_mac_mubuff_v2()

```
she_err_t she_generate_fast_mac_mubuff_v2 (
    she_hdl_t utils_handle,
    op_generate_mac_t * args )
```

Generates a MAC (V2) of a given message (supports length greater than 16 bytes) with the help of a key identified by key_id.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.4 CMD_VERIFY_MAC

Data Structures

- struct [op_verify_mac_t](#)

Macros

- #define [SHE_FAST_MAC_FLAGS_VERIFICATION](#) 1
Flag to verify MAC.
- #define [SHE_FAST_MAC_FLAGS_VERIF_BIT_LEN](#) 2
Flag to verify variable length MAC.
- #define [SHE_FAST_MAC_VERIFICATION_STATUS_OK](#) 0x5a3cc3a5
verification status of MAC
- #define [MAC_BYTES_LENGTH](#) 0
MAC length expressed in bytes.
- #define [MAC_BITS_LENGTH](#) 1
MAC length expressed in bits.
- #define [SHE_MAC_VERIFICATION_SUCCESS](#) 0
indication of mac verification success
- #define [SHE_MAC_VERIFICATION_FAILED](#) 1
indication of mac verification failure

Functions

- [she_err_t she_verify_mac](#) ([she_hdl_t](#) utils_handle, [op_verify_mac_t](#) *args)
- [she_err_t she_verify_fast_mac_mubuff_v2](#) ([she_hdl_t](#) utils_handle, [op_verify_mac_t](#) *args)

5.4.1 Detailed Description

SHE supports two types of functions to verify MAC. 1: for 16 bytes message 2: V2-for 16 or greater than 16 bytes message.

5.4.2 Data Structure Documentation

5.4.2.1 struct op_verify_mac_t

Structure describing the FAST MAC generation operation arguments

Data Fields

| | | |
|-----------|---------------------|---|
| uint8_t | flags | optional flag to identify the operation(generate/verify) |
| uint16_t | key_ext | identifier of the key extension to be used for the operation |
| uint16_t | key_id | identifier of the key to be used for the operation |
| uint16_t | message_length | length in bytes of the input message. The message is padded to be a multiple of 128 bits by SHE |
| uint8_t * | message | pointer to the message to be processed |
| uint8_t * | mac | pointer to the MAC to be compared |
| uint8_t | mac_length | number of MAC bytes to be compared with the expected value. It cannot be lower than 4 bytes. |
| uint32_t | verification_status | result of the MAC comparison |
| uint8_t | mac_length_encoding | mac length expressed in bytes or bits |

5.4.3 Function Documentation

5.4.3.1 she_verify_mac()

```
she_err_t she_verify_mac (
    she_hdl_t utils_handle,
    op_verify_mac_t * args )
```

Verify the MAC of a given message with the help of a key identified by key_id.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.4.3.2 she_verify_fast_mac_mubuff_v2()

```
she_err_t she_verify_fast_mac_mubuff_v2 (
    she_hdl_t utils_handle,
    op_verify_mac_t * args )
```

Verify the MAC (V2) of a given message (supports message length greater than 16 bytes) with the help of a key identified by key_id.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.5 CMD_GET_ID

Data Structures

- struct [op_get_id_args_t](#)

Macros

- #define [SHE_CHALLENGE_SIZE](#) 16u
size of the input challenge vector is 128 bits.
- #define [SHE_ID_SIZE](#) 15u
size of the Identity(ID) returned is 120 bits.
- #define [SHE_MAC_SIZE](#) 16u
size of the computed MAC is 128 bits.

Functions

- [she_err_t she_get_id](#) ([she_hdl_t](#) utils_handle, [op_get_id_args_t](#) *args)

5.5.1 Detailed Description

Status register can be read using below command, its an 8 bit register and has very useful information (more bit description, please refer SHE specification)

5.5.2 Data Structure Documentation

5.5.2.1 struct op_get_id_args_t

Structure describing the GET ID operation arguments

Data Fields

| | | |
|---------|---|---|
| uint8_t | challenge[SHE_CHALLENGE_SIZE] | input Challenge vector |
| uint8_t | id[SHE_ID_SIZE] | identity (UID) returned by the command |
| uint8_t | sreg | status register returned by the command |
| uint8_t | mac[SHE_MAC_SIZE] | MAC returned by the command. |

5.5.3 Function Documentation

5.5.3.1 she_get_id()

```
she_err_t she_get_id (
    she_hdl_t utils_handle,
```

```
op_get_id_args_t * args )
```

This function returns the identity (UID) and the value of the status register protected by a MAC over a challenge and the data. User can call this function only after opening the utility service.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.6 SHE Get Info

Get miscellaneous information.

Data Structures

- struct [op_get_info_args_t](#)

Functions

- [she_err_t she_get_info](#) ([she_hdl_t](#) session_hdl, [op_get_info_args_t](#) *args)

5.6.1 Detailed Description

Get miscellaneous information.

This function return, among others, all the information needed to build a valid signed message.

5.6.2 Data Structure Documentation

5.6.2.1 struct op_get_info_args_t

Structure describing the get info operation member arguments

Data Fields

| | | |
|-----------|------------------------|--|
| uint32_t | user_sab_id | Stores User identifier (32bits) |
| uint8_t * | chip_unique_id | Stores the chip unique identifier. |
| uint16_t | chip_unq_id_sz | Size of the chip unique identifier in bytes. |
| uint16_t | chip_monotonic_counter | Stores the chip monotonic counter value (16bits) |
| uint16_t | chip_life_cycle | Stores the chip current life cycle bitfield (16bits) |
| uint32_t | version | Stores the module version (32bits) |
| uint32_t | version_ext | Stores the module extended version (32bits) |
| uint8_t | fips_mode | Stores the FIPS mode bitfield (8bits). Bitmask definition: bit0 - FIPS mode of operation: <ul style="list-style-type: none"> • value 0 - part is running in FIPS non-approved mode. • value 1 - part is running in FIPS approved mode. bit1 - FIPS certified part: <ul style="list-style-type: none"> • value 0 - part is not FIPS certified. • value 1 - part is FIPS certified. bit2-7: reserved <ul style="list-style-type: none"> • value 0. |

5.6.3 Function Documentation

5.6.3.1 she_get_info()

```
she_err_t she_get_info (
    she_hdl_t session_hdl,
    op_get_info_args_t * args )
```

User can call this function only after having opened the SHE session.

Parameters

| | |
|--------------------|---|
| <i>session_hdl</i> | handle identifying the active session. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.7 SHE Commands

Modules

- [CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB](#)
- [CMD_EXPORT_RAM_KEY](#)
- [CMD_GENERATE_MAC](#)
- [CMD_VERIFY_MAC](#)
- [CMD_GET_ID](#)
- [CMD_GET_STATUS](#)
- [CMD_LOAD_KEY](#)
- [CMD_LOAD_PLAIN_KEY](#)
- [CMD_INIT_RNG](#)
- [CMD_RND](#)
- [CMD_EXTEND_SEED](#)
- [last rating code](#)
- [CMD_CANCEL](#)

5.7.1 Detailed Description

5.8 CMD_GET_STATUS

Data Structures

- struct [op_get_status_args_t](#)

Functions

- [she_err_t she_get_status](#) ([she_hdl_t](#) *utils_handle*, [op_get_status_args_t](#) *args)

5.8.1 Detailed Description

Return the content of status register

5.8.2 Data Structure Documentation

5.8.2.1 struct op_get_status_args_t

Structure describing the get status operation arguments

Data Fields

| | | |
|---------|--------|----------------------|
| uint8_t | sreg | status register bits |
| uint8_t | pad[3] | padding bytes |

5.8.3 Function Documentation

5.8.3.1 she_get_status()

```
she_err_t she_get_status (
    she_hdl_t  utils_handle,
    op_get_status_args_t * args )
```

Command to get the content of the status register

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code.

5.9 Session

SHE session is the first session to be opened, all other sessions are based on it.

Data Structures

- struct [she_session_hdl_s](#)
- struct [she_service_hdl_s](#)
- struct [open_session_args_t](#)

Macros

- #define [SHE_HANDLE_NONE](#) (0x0)
Handle not available.
- #define [SHE_MAX_SESSIONS](#) (8u)
Maximum sessions supported.
- #define [SHE_MAX_SERVICES](#) (32u)
Maximum services supported.
- #define [MAX_KEY_STORE_SESSIONS](#) (5u)
Maximum Key store sessions supported.
- #define [SHE_OPEN_SESSION_PRIORITY_LOW](#) (0x00U)
Low priority session, default setting on platforms that doesn't support sessions priorities.
- #define [SHE_OPEN_SESSION_PRIORITY_HIGH](#) (0x01U)
High Priority session.
- #define [SHE_OPEN_SESSION_FIPS_MODE_MASK](#) BIT(0)
Only FIPS certified operations authorized in this session.
- #define [SHE_OPEN_SESSION_EXCLUSIVE_MASK](#) BIT(1)
No other SHE session will be authorized on the same security enclave.
- #define [SHE_OPEN_SESSION_LOW_LATENCY_MASK](#) BIT(3)
Use a low latency SHE implementation.
- #define [SHE_OPEN_SESSION_NO_KEY_STORE_MASK](#) BIT(4)
No key store will be attached to this session. May provide better performances on some operation depending on the implementation. Usage of the session will be restricted to operations that doesn't involve secret keys (e.g. random generation)

Typedefs

- typedef uint32_t [she_hdl_t](#)

Functions

- struct [she_session_hdl_s](#) * [she_session_hdl_to_ptr](#) (uint32_t hdl)
- void [delete_she_session](#) (struct [she_session_hdl_s](#) *s_ptr)
- struct [she_session_hdl_s](#) * [add_she_session](#) (void)
- struct [she_service_hdl_s](#) * [she_service_hdl_to_ptr](#) (uint32_t hdl)
- void [delete_she_service](#) (struct [she_service_hdl_s](#) *s_ptr)
- struct [she_service_hdl_s](#) * [add_she_service](#) (struct [she_session_hdl_s](#) *session)
- [she_err_t](#) [she_open_session](#) ([open_session_args_t](#) *args, [she_hdl_t](#) *session_hdl)
- [she_err_t](#) [she_close_session](#) ([she_hdl_t](#) session_hdl)

5.9.1 Detailed Description

SHE session is the first session to be opened, all other sessions are based on it.

All functions and related data types required for opening a SHE session are described below.

5.9.2 Data Structure Documentation

5.9.2.1 struct she_session_hdl_s

Structure describing the session handle members

Data Fields

| | | |
|--------------------------|-------------|--------------------------------------|
| struct plat_os_abs_hdl * | phdl | Pointer to OS device node. |
| uint32_t | session_hdl | Session handle. |
| uint32_t | mu_type | Session MU type. |
| uint32_t | last_rating | last error code returned by command. |

5.9.2.2 struct she_service_hdl_s

Structure describing the service handle members

Data Fields

| | | |
|--|-------------|----------------------------|
| struct she_session_hdl_s * | session | Pointer to session handle. |
| uint32_t | service_hdl | Service handle. |

5.9.2.3 struct open_session_args_t

Structure detailing the open session operation member arguments

Data Fields

| | | |
|----------|------------------|---|
| uint32_t | session_hdl | Session handle. |
| uint32_t | mu_type | MU type on which session will be opened. |
| uint8_t | session_priority | Priority of the operations performed in this session. |
| uint8_t | operating_mode | Options for the session to be opened (bitfield). |
| uint8_t | interrupt_idx | Interrupt number of the MU used to indicate data availability. |
| uint8_t | mu_id | index of the MU as per PLAT point of view. |
| uint8_t | tz | Trust Zone (secure) flag - Reflects the XRDC configuration on the MU. |
| uint8_t | did | DID associated with the MU in the XRDC configuration. |

5.9.3 Typedef Documentation

5.9.3.1 she_hdl_t

```
typedef uint32_t she_hdl_t
```

Define the SHE handle type

5.9.4 Function Documentation

5.9.4.1 she_session_hdl_to_ptr()

```
struct she_session_hdl_s* she_session_hdl_to_ptr (
    uint32_t hdl )
```

Returns pointer to the session handle

Parameters

| | |
|------------|---------------------------------|
| <i>hdl</i> | identifying the session handle. |
|------------|---------------------------------|

Returns

pointer to the session handle.

5.9.4.2 delete_she_session()

```
void delete_she_session (
    struct she_session_hdl_s * s_ptr )
```

Delete the session

Parameters

| | |
|--------------|----------------------------------|
| <i>s_ptr</i> | pointer identifying the session. |
|--------------|----------------------------------|

5.9.4.3 add_she_session()

```
struct she_session_hdl_s* add_she_session (
    void )
```

Add the session

Returns

pointer to the session.

5.9.4.4 she_service_hdl_to_ptr()

```
struct she_service_hdl_s* she_service_hdl_to_ptr (
    uint32_t hdl )
```

Returns pointer to the service handle

Parameters

| | |
|------------|---------------------------------|
| <i>hdl</i> | identifying the session handle. |
|------------|---------------------------------|

Returns

pointer to the service handle.

5.9.4.5 delete_she_service()

```
void delete_she_service (
    struct she_service_hdl_s * s_ptr )
```

Delete the service

Parameters

| | |
|--------------|----------------------------------|
| <i>s_ptr</i> | pointer identifying the service. |
|--------------|----------------------------------|

5.9.4.6 add_she_service()

```
struct she_service_hdl_s* add_she_service (
    struct she_session_hdl_s * session )
```

Add the service

Returns

pointer to the service.

5.9.4.7 she_open_session()

```
she_err_t she_open_session (
    open_session_args_t * args,
    she_hdl_t * session_hdl )
```

Parameters

| | |
|--------------------|---|
| <i>args</i> | pointer to the structure containing the function arguments. |
| <i>session_hdl</i> | pointer to where the session handle must be written. |

Returns

error code.

5.9.4.8 she_close_session()

```
she_err_t she_close_session (
    she_hdl_t session_hdl )
```

Terminate a previously opened session. All the services opened under this session are closed as well

Parameters

| | |
|--------------------|---|
| <i>session_hdl</i> | pointer to the handle identifying the session to be closed. |
|--------------------|---|

Returns

error code.

5.10 SHE keys

Identifiers for SHE keys.

Macros

- `#define SHE_KEY_1 (0x04)`
- `#define SHE_KEY_2 (0x05)`
- `#define SHE_KEY_3 (0x06)`
- `#define SHE_KEY_4 (0x07)`
- `#define SHE_KEY_5 (0x08)`
- `#define SHE_KEY_6 (0x09)`
- `#define SHE_KEY_7 (0x0a)`
- `#define SHE_KEY_8 (0x0b)`
- `#define SHE_KEY_9 (0x0c)`
- `#define SHE_KEY_10 (0x0d)`
- `#define SHE_RAM_KEY (0x0e)`
- `#define SHE_KEY_SIZE_IN_BYTES 16u`
SHE keys are 128 bits (16 bytes) long.
- `#define M1_M3_M5_KEY_SIZE_IN_WORDS (SHE_KEY_SIZE_IN_BYTES >> 2)`
size of M1, M3 ad M5 in words.
- `#define M2_M4_KEY_SIZE_IN_WORDS (SHE_KEY_SIZE_IN_BYTES >> 1)`
size of M2 ad M4 in words.

5.10.1 Detailed Description

Identifiers for SHE keys.

Refer SHE specification for more information.

5.11 SHE+ key extension

Identifiers for the SHE key extension.

Macros

- #define `SHE_KEY_DEFAULT` (0x00)
no key extension: keys from 0 to 10 as defined in SHE specification.
- #define `SHE_KEY_N_EXT_1` (0x10)
keys 11 to 20.
- #define `SHE_KEY_N_EXT_2` (0x20)
keys 21 to 30.
- #define `SHE_KEY_N_EXT_3` (0x30)
keys 31 to 40.
- #define `SHE_KEY_N_EXT_4` (0x40)
keys 41 to 50.

5.11.1 Detailed Description

Identifiers for the SHE key extension.

There are 5 SHE key stores in i.MX95 and 1 SHE keystore in i.MX8DXL

Each key store contains:

- 1 SECRET KEY (id = 0x0)
- 1 MASTER ECU KEY (id = 0x1)
- 1 BOOT MAC KEY (id = 0x2)
- 1 BOOT MAC (id = 0x3)
- 10 KEY SLOTS (id = 0x04 to 0xD)
- 40 extra KEY SLOTS(id = keyID | key_ext,
 - keyID from 0x04 to 0xD,
 - key_ext like below picture)
- 1 RAM KEY (id = 0xE)

5.12 Key store

User must open a key store service flow in order to perform all operations on keys.

Data Structures

- struct [open_svc_key_store_args_t](#)
- struct [op_key_store_reprov_en_args_t](#)

Macros

- #define [KEY_STORE_OPEN_FLAGS_DEFAULT](#) 0x0u
default flags
- #define [KEY_STORE_OPEN_FLAGS_CREATE](#) 0x1u
Create a key store.
- #define [KEY_STORE_OPEN_FLAGS_SHE](#) 0x2u
Target key store is a SHE key store.
- #define [KEY_STORE_OPEN_FLAGS_SET_MAC_LEN](#) 0x8u
Check min mac length.
- #define [KEY_STORE_OPEN_FLAGS_SHARED](#) 0x20u
Target key store is a shared key store (applicable only for i.MX95)
- #define [KEY_STORE_OPEN_FLAGS_STRICT_OPERATION](#) 0x80u
The request is completed only when the key store has been written in the NVM and the monotonic counter has been updated. This flag is applicable for CREATE operation only.
- #define [SHE_STORAGE_MIN_MAC_BIT_LENGTH_DEFAULT](#) 32u
default MAC verification length in bits

Functions

- [she_err_t she_open_key_store_service](#) ([she_hdl_t](#) session_hdl, [open_svc_key_store_args_t](#) *args)
- [she_err_t she_close_key_store_service](#) ([she_hdl_t](#) key_store_handle)

5.12.1 Detailed Description

User must open a key store service flow in order to perform all operations on keys.

- create a new key store
- perform operations involving keys stored in the key store (ciphering, MAC generation/verification...)
- perform a key store reprovisioning using a signed message. A key store re-provisioning results in erasing all the key stores handled by the SHE.

To grant access to the key store, the caller is authenticated against the domain ID (DID) and Messaging Unit used at the keystore creation, additionally an authentication nonce can be provided.

5.12.2 Data Structure Documentation

5.12.2.1 struct open_svc_key_store_args_t

Structure specifying the open key store service member arguments

Data Fields

| | | |
|-----------|----------------------|--|
| uint32_t | key_store_hdl | handle identifying the key store service flow |
| uint32_t | key_store_identifier | user defined id identifying the key store. Only one key store service can be opened on a given key_store_identifier. |
| uint32_t | authentication_nonce | user defined nonce used as authentication proof for accessing the key store. |
| uint8_t | flags | bitmap specifying the services properties. |
| uint16_t | max_updates_number | maximum number of updates authorized for the key store. <ul style="list-style-type: none"> Valid only for create operation. This parameter has the goal to limit the occupation of the monotonic counter used as anti-rollback protection. If the maximum number of updates is reached, HSM still allows key store updates but without updating the monotonic counter giving the opportunity for rollback attacks. |
| uint8_t | min_mac_length | it corresponds to the minimum mac length (in bits) accepted to perform MAC verification operations. Only used upon key store creation when KEY_STORE_FLAGS_SET_MAC_LEN bit is set. It is effective only for MAC verification operations with the mac length expressed in bits. It can be used to replace the default value (32 bits). It impacts all MAC algorithms and all key lengths. It must be different from 0. When in FIPS approved mode values < 32 bits are not allowed. Only used on devices implementing SECO FW. |
| uint8_t * | signed_message | pointer to signed_message to be sent only in case of key store re-provisioning. |
| uint16_t | signed_msg_size | size of the signed_message to be sent only in case of key store re-provisioning. |

5.12.2.2 struct op_key_store_reprov_en_args_t

Structure describing the key store reprovisioning enable operation arguments

Data Fields

| | | |
|-----------|-----------------|--------------------------------------|
| uint8_t * | signed_message | signed content payload |
| uint32_t | signed_msg_size | signed content payload size in bytes |

5.12.3 Function Documentation

5.12.3.1 she_open_key_store_service()

```
she_err_t she_open_key_store_service (
    she_hdl_t session_hdl,
    open_svc_key_store_args_t * args )
```

Open a service flow on the specified key store.

Parameters

| | |
|--------------------|---|
| <i>session_hdl</i> | SHE handle identifying the current session. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code.

5.12.3.2 she_close_key_store_service()

```
she_err_t she_close_key_store_service (
    she_hdl_t key_store_handle )
```

Terminate a previously opened key store service flow

Parameters

| | |
|-------------------------|---|
| <i>key_store_handle</i> | handle identifying the key store service. |
|-------------------------|---|

Returns

error code.

5.13 CMD_LOAD_KEY

Data Structures

- struct [op_key_update_args_t](#)
- struct [op_key_update_ext_args_t](#)

Macros

- #define [SHE_LOAD_KEY_EXT_FLAGS_STRICT_OPERATION](#) BIT(7)
User can use this flag to perform multiple updates before writing the key store into the NVM and incrementing the monotonic counter.

Functions

- [she_err_t she_key_update](#) ([she_hdl_t](#) utils_handle, [op_key_update_args_t](#) *args)
- [she_err_t she_key_update_ext](#) ([she_hdl_t](#) utils_handle, [op_key_update_ext_args_t](#) *args)

5.13.1 Detailed Description

Supports two variations of KEY update.

5.13.2 Data Structure Documentation

5.13.2.1 struct op_key_update_args_t

Structure describing the key update operation arguments

Data Fields

| | | |
|-----------|--------------|--|
| uint32_t | utils_handle | Handle to utils service. |
| uint32_t | key_ext | identifier of the key extension to be used for the operation |
| uint32_t | key_id | identifier of the key to be used for the operation |
| uint8_t * | m1 | pointer to M1 message |
| uint8_t | m1_size | size of M1 message - 128 bits |
| uint8_t * | m2 | pointer to M2 message |
| uint8_t | m2_size | size of M2 message - 256 bits |
| uint8_t * | m3 | pointer to M3 message |
| uint8_t | m3_size | size of M3 message - 128 bits |
| uint8_t * | m4 | pointer to the output address for M4 message |
| uint8_t | m4_size | size of M4 message - 256 bits |
| uint8_t * | m5 | pointer to the output address for M5 message |
| uint8_t | m5_size | size of M5 message - 128 bits |

5.13.2.2 struct op_key_update_ext_args_t

Structure describing the key update extension operation arguments

Data Fields

| | | |
|-----------|--------------|--|
| uint32_t | utils_handle | Handle to utils service. |
| uint32_t | key_ext | identifier of the key extension to be used for the operation |
| uint32_t | key_id | identifier of the key to be used for the operation |
| uint8_t * | m1 | pointer to M1 message |
| uint8_t | m1_size | size of M1 message - 128 bits |
| uint8_t * | m2 | pointer to M2 message |
| uint8_t | m2_size | size of M2 message - 256 bits |
| uint8_t * | m3 | pointer to M3 message |
| uint8_t | m3_size | size of M3 message - 128 bits |
| uint8_t * | m4 | pointer to the output address for M4 message |
| uint8_t | m4_size | size of M4 message - 256 bits |
| uint8_t * | m5 | pointer to the output address for M5 message |
| uint8_t | m5_size | size of M5 message - 128 bits |
| uint8_t | flags | bitmap specifying the operations property |

5.13.3 Function Documentation

5.13.3.1 she_key_update()

```
she_err_t she_key_update (
    she_hdl_t utils_handle,
    op_key_update_args_t * args )
```

Update an internal key of SHE based on memory update protocol refer specification for more information. The request is completed only when the new key has been written in the NVM. The monotonic counter is incremented for each successful update.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.13.3.2 she_key_update_ext()

```
she_err_t she_key_update_ext (
    she_hdl_t utils_handle,
    op_key_update_ext_args_t * args )
```

This is an extension of the CMD_LOAD_KEY. The functionality of the CMD_LOAD_KEY is extended by adding a flag argument. The updates to the key store must be considered as effective only after an operation specifying the flag "STRICT OPERATION" is acknowledged by SHE.

The request is completed only when the key store is written in the NVM and the monotonic counter is incremented.

Parameters

| | |
|---------------------|---|
| <i>utils_handle</i> | handle identifying the utils service |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.14 CMD_LOAD_PLAIN_KEY

Data Structures

- struct [op_load_plain_key_args_t](#)

Functions

- [she_err_t she_load_plain_key](#) ([she_hdl_t](#) *utils_handle*, [op_load_plain_key_args_t](#) *args)

5.14.1 Detailed Description

Key is handed over in plaintext. A plain key can only be loaded into the RAM_KEY slot.

5.14.2 Data Structure Documentation

5.14.2.1 struct op_load_plain_key_args_t

Structure describing the plain key load operation arguments

Data Fields

| | | |
|-------------------------|--|----------------------|
| uint8_t | key[SHE_KEY_SIZE_IN_BYTES] | pointer to plain key |
|-------------------------|--|----------------------|

5.14.3 Function Documentation

5.14.3.1 she_load_plain_key()

```
she_err_t she_load_plain_key (  
    she_hdl_t utils_handle,  
    op_load_plain_key_args_t * args )
```

Load a key as plaintext to the RAM_KEY slot without encryption and verification.

Parameters

| | |
|---------------------|--|
| <i>utils_handle</i> | pointer to the SHE utils handle |
| <i>args</i> | pointer to structure containg function arguments |

Returns

error code

5.15 CMD_INIT_RNG

Functions

- [she_err_t she_open_rng_service](#) ([she_hdl_t](#) session_hdl, [open_svc_rng_args_t](#) *args)
- [she_err_t she_close_rng_service](#) ([she_hdl_t](#) rng_handle)

5.15.1 Detailed Description

We need to open RNG service before generating Random numbers after every power cycle/reset.

5.15.2 Function Documentation

5.15.2.1 she_open_rng_service()

```
she_err_t she_open_rng_service (  
    she_hdl_t session_hdl,  
    open_svc_rng_args_t * args )
```

Initializes the seed and derives a key for the PRNG.

User can call this function only after having opened a session.

Parameters

| | |
|--------------------|---|
| <i>session_hdl</i> | handle identifying the current SHE session. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.15.2.2 she_close_rng_service()

```
she_err_t she_close_rng_service (  
    she_hdl_t rng_handle )
```

Terminate a previously opened rng service flow

Parameters

| | |
|-------------------|-------------------------------------|
| <i>rng_handle</i> | handle identifying the RNG session. |
|-------------------|-------------------------------------|

Returns

error code

5.16 CMD_RND

Data Structures

- struct [open_svc_rng_args_t](#)
- struct [op_get_random_args_t](#)

Macros

- #define [SHE_RND_SIZE](#) 16u
size of random data

Typedefs

- typedef uint8_t [svc_rng_flags_t](#)

Functions

- [she_err_t she_get_random](#) ([she_hdl_t](#) rng_handle, [op_get_random_args_t](#) *args)

5.16.1 Detailed Description

The random number generator has to be initialized by CMD_INIT_RNG before random numbers can be supplied.

5.16.2 Data Structure Documentation

5.16.2.1 struct open_svc_rng_args_t

Structure detailing session open operation member arguments

Data Fields

| | | |
|---------------------------------|-------------|---|
| svc_rng_flags_t | flags | bitmap indicating the service flow properties |
| uint8_t | reserved[3] | reserved bits |
| uint32_t | rng_hdl | rng handle |

5.16.2.2 struct op_get_random_args_t

Structure detailing the get random number operation member arguments

Data Fields

| | | |
|---------------------------|-------------|--|
| uint8_t * | output | pointer to the output area where the random number must be written |
| uint32_t | random_size | length in bytes of the random number to be provided. |

Data Fields

| | | |
|-----------------|-------------|---|
| svc_rng_flags_t | svc_flags | bitmap indicating the service flow properties |
| uint8_t | reserved[3] | reserved bits |

5.16.3 Function Documentation

5.16.3.1 she_get_random()

```
she_err_t she_get_random (
    she_hdl_t rng_handle,
    op_get_random_args_t * args )
```

returns a vector of 128 random bits.

Parameters

| | |
|-------------------|---|
| <i>rng_handle</i> | handle identifying the RNG service |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code

5.17 CMD_EXTEND_SEED

Data Structures

- struct [op_rng_extend_seed_t](#)

Macros

- #define [SHE_ENTROPY_SIZE](#) 16u
size of entropy for SHE

Functions

- [she_err_t](#) [she_extend_seed](#) ([she_hdl_t](#) rng_handle, [op_rng_extend_seed_t](#) *args)

5.17.1 Detailed Description

The random number generator has to be initialized by CMD_INIT_RNG before the seed can be extended.

5.17.2 Data Structure Documentation

5.17.2.1 struct op_rng_extend_seed_t

Structure describing the RNG extend seed operation arguments

Data Fields

| | | |
|----------|--------------|------------------------|
| uint32_t | entropy[4] | entropy to extend seed |
| uint32_t | entropy_size | entropy size |

5.17.3 Function Documentation

5.17.3.1 she_extend_seed()

```
she_err_t she_extend_seed (
    she_hdl_t rng_handle,
    op_rng_extend_seed_t * args )
```

extends the seed of the PRNG by compressing the former seed value and the supplied entropy into a new seed which will be used to generate the following random numbers.

Parameters

| | |
|-------------------|--|
| <i>rng_handle</i> | handle identifying the RNG service |
| <i>args</i> | pointer to the structure containing entropy vector (128bits) |

Returns

error code

5.18 Shared Buffer

Shared Memory and shared buffer.

Data Structures

- struct [op_shared_buf_args_t](#)

5.18.1 Detailed Description

Shared Memory and shared buffer.

This is applicable for SECO only not valid for V2X (i.MX95)

5.18.2 Data Structure Documentation

5.18.2.1 struct op_shared_buf_args_t

Structure describing the get shared buffer operation arguments

Data Fields

| | | |
|----------|-------------------|--|
| uint16_t | shared_buf_offset | offset of the shared buffer in secure memory |
| uint16_t | shared_buf_size | size in bytes of the allocated shared buffer |

5.19 Error codes

Error codes returned by SHE functions.

Macros

- `#define SAB_INVALID_MESSAGE 0x01`
Invalid/Unknown message.
- `#define SAB_INVALID_ADDRESS 0x02`
Invalid Address.
- `#define SAB_UNKNOWN_ID 0x03`
Unknown Id.
- `#define SAB_INVALID_PARAM 0x04`
MU sanity check failed / Invalid parameters.
- `#define SAB_NVM_ERROR 0x05`
NVM general error.
- `#define SAB_OUT_OF_MEMORY 0x06`
Internal memory allocation failed.
- `#define SAB_UNKNOWN_HANDLE 0x07`
Unknown handle.
- `#define SAB_UNKNOWN_KEY_STORE 0x08`
Key store with provided key store ID does not exist (load operation).
- `#define SAB_KEY_STORE_AUTH 0x09`
A key store authentication is failing.
- `#define SAB_KEY_STORAGE_ERROR 0x0A`
Key store creation/load failure.
- `#define SAB_ID_CONFLICT 0x0B`
A Key store using the same key id already exists (create operation).
- `#define SAB_RNG_NOT_STARTED 0x0C`
Internal RNG not started.
- `#define SAB_CMD_NOT_SUPPORTED 0x0D`
Functionality not supported on current service configuration.
- `#define SAB_INVALID_LIFECYCLE 0x0E`
Invalid lifecycle for requested operation.
- `#define SAB_KEY_STORE_CONFLICT 0x0F`
The key store already exists (load operation).
- `#define SAB_KEY_STORE_COUNTER 0x10`
Issue occurred while updating the key store counter.
- `#define SAB_FEATURE_NOT_SUPPORTED 0x11`
Feature is not supported.
- `#define SAB_SELF_TEST_FAILURE 0x12`
Self test execution failed.
- `#define SAB_NOT_READY 0x13`
System not ready to accept service request.
- `#define SAB_FEATURE_DISABLED 0x14`
Feature disabled.
- `#define SAB_FATAL_FAILURE 0xFF`
fatal error

Enumerations

```

• enum she_err_t {
    SHE_NO_ERROR = 0x00,
    SHE_INVALID_MESSAGE = SAB_INVALID_MESSAGE,
    SHE_INVALID_ADDRESS = SAB_INVALID_ADDRESS,
    SHE_UNKNOWN_ID = SAB_UNKNOWN_ID,
    SHE_INVALID_PARAM = SAB_INVALID_PARAM,
    SHE_NVM_ERRO = SAB_NVM_ERROR,
    SHE_OUT_OF_MEMORY = SAB_OUT_OF_MEMORY,
    SHE_UNKNOWN_HANDLE = SAB_UNKNOWN_HANDLE,
    SHE_UNKNOWN_KEY_STORE = SAB_UNKNOWN_KEY_STORE,
    SHE_KEY_STORE_AUTH = SAB_KEY_STORE_AUTH,
    SHE_KEY_STORAGE_ERROR = SAB_KEY_STORAGE_ERROR,
    SHE_ID_CONFLICT = SAB_ID_CONFLICT,
    SHE_RNG_NOT_STARTED = SAB_RNG_NOT_STARTED,
    SHE_CMD_NOT_SUPPORTED = SAB_CMD_NOT_SUPPORTED,
    SHE_INVALID_LIFECYCLE = SAB_INVALID_LIFECYCLE,
    SHE_KEY_STORE_CONFLICT = SAB_KEY_STORE_CONFLICT,
    SHE_KEY_STORE_COUNTER = SAB_KEY_STORE_COUNTER,
    SHE_FEATURE_NOT_SUPPORTED = SAB_FEATURE_NOT_SUPPORTED,
    SHE_SELF_TEST_FAILURE = SAB_SELF_TEST_FAILURE,
    SHE_NOT_READY = SAB_NOT_READY,
    SHE_FEATURE_DISABLED = SAB_FEATURE_DISABLED,
    SHE_UNKNOWN_WARNING = 0x27,
    SHE_SEQUENCE_ERROR_RATING = 0xD1,
    SHE_KEY_NOT_AVAILABLE_RATING = 0xD2,
    SHE_KEY_INVALID_RATING = 0xD3,
    SHE_KEY_EMPTY_RATING = 0xD4,
    SHE_NO_SECURE_BOOT_RATING = 0xD5,
    SHE_KEY_WRITE_PROTECTED_RATING = 0xD6,
    SHE_KEY_UPDATE_ERROR_RATING = 0xD7,
    SHE_RNG_SEED_RATING = 0xD8,
    SHE_NO_DEBUGGING_RATING = 0xD9,
    SHE_BUSY_RATING = 0xDA,
    SHE_MEMORY_FAILURE_RATING = 0xDB,
    SHE_GENERAL_ERROR = 0xDC,
    SHE_LIB_ERROR = 0xEF,
    SHE_FATAL_FAILURE = SAB_FATAL_FAILURE }

```

5.19.1 Detailed Description

Error codes returned by SHE functions.

Details on all error codes returned by SHE APIs

5.19.2 Enumeration Type Documentation

5.19.2.1 she_err_t

```
enum she_err_t
```

Error codes returned by SHE functions.

Enumerator

| | |
|--------------------------------|---|
| SHE_NO_ERROR | Success. |
| SHE_INVALID_MESSAGE | Invalid/Unknown message. |
| SHE_INVALID_ADDRESS | Invalid Address. |
| SHE_UNKNOWN_ID | Unknown Id. |
| SHE_INVALID_PARAM | MU sanity check failed / Invalid parameters. |
| SHE_NVM_ERRO | NVM general error. |
| SHE_OUT_OF_MEMORY | Internal memory allocation failed. |
| SHE_UNKNOWN_HANDLE | Unknown handle. |
| SHE_UNKNOWN_KEY_STORE | Key store with provided key store ID does not exist (load operation). |
| SHE_KEY_STORE_AUTH | A key store authentication is failing. |
| SHE_KEY_STORAGE_ERROR | Key store creation/load failure. |
| SHE_ID_CONFLICT | A Key store using the same key id already exists (create operation). |
| SHE_RNG_NOT_STARTED | Internal RNG not started. |
| SHE_CMD_NOT_SUPPORTED | Functionality not supported on current service configuration. |
| SHE_INVALID_LIFECYCLE | Invalid lifecycle for requested operation. |
| SHE_KEY_STORE_CONFLICT | The key store already exists (load operation). |
| SHE_KEY_STORE_COUNTER | Issue occurred while updating the key store counter. |
| SHE_FEATURE_NOT_SUPPORTED | Feature is not supported. |
| SHE_SELF_TEST_FAILURE | Self test execution failed. |
| SHE_NOT_READY | System not ready to accept service request. |
| SHE_FEATURE_DISABLED | Feature disabled. |
| SHE_UNKNOWN_WARNING | SHE Unknown Warning. |
| SHE_SEQUENCE_ERROR_RATING | Invalid sequence of commands. |
| SHE_KEY_NOT_AVAILABLE_RATING | Key is locked. |
| SHE_KEY_INVALID_RATING | Key not allowed for the given operation. |
| SHE_KEY_EMPTY_RATING | Key has not been initialized yet. |
| SHE_NO_SECURE_BOOT_RATING | Conditions for a secure boot process are not met. |
| SHE_KEY_WRITE_PROTECTED_RATING | Memory slot for this key has been write-protected. |
| SHE_KEY_UPDATE_ERROR_RATING | Key update did not succeed, errors in verification of message. |
| SHE_RNG_SEED_RATING | The seed has not been initialized. |
| SHE_NO_DEBUGGING_RATING | Internal debugging is not possible. |
| SHE_BUSY_RATING | SHE is busy. |
| SHE_MEMORY_FAILURE_RATING | Memory Error. |
| SHE_GENERAL_ERROR | SHE General error. |
| SHE_LIB_ERROR | SHE library error. |
| SHE_FATAL_FAILURE | fatal error |

5.20 Utils

The SHE utils service flow can be opened only after opening SHE key storage handle.

Data Structures

- struct [op_open_utils_args_t](#)

Functions

- [she_err_t she_open_utils](#) ([she_hdl_t](#) key_store_handle, [op_open_utils_args_t](#) *args)
- [she_err_t she_close_utils](#) ([she_hdl_t](#) utils_handle)

5.20.1 Detailed Description

The SHE utils service flow can be opened only after opening SHE key storage handle.

User must open a SHE utils service flow in order to perform all operations:

- Create a utils handle
- perform SHE key update extension
- update SHE plain key
- export SHE plain key
- get SHE identity (UID)
- get SHE status register
- perform MAC generation and verification in fast mode for a SHE session

5.20.2 Data Structure Documentation

5.20.2.1 struct [op_open_utils_args_t](#)

Structure describing the open utils service operation arguments

Data Fields

| | | |
|--------------------------|------------------------------|--|
| uint32_t | utils_handle | |
|--------------------------|------------------------------|--|

5.20.3 Function Documentation

5.20.3.1 she_open_utils()

```
she_err_t she_open_utils (
    she_hdl_t key_store_handle,
    op_open_utils_args_t * args )
```

Open SHE utils service flow on the specified key store. The SHE utils service flow can be opened only after opening SHE key storage handle.

Parameters

| | |
|-------------------------|---|
| <i>key_store_handle</i> | handle identifying the key store service. |
| <i>args</i> | pointer to the structure containing the function arguments. |

Returns

error code.

5.20.3.2 she_close_utils()

```
she_err_t she_close_utils (
    she_hdl_t utils_handle )
```

Terminate a previously opened utils service flow

Parameters

| | |
|---------------------|---------------------------------------|
| <i>utils_handle</i> | handle identifying the utils service. |
|---------------------|---------------------------------------|

Returns

error code.

5.21 last rating code

Functions

- uint32_t [she_get_last_rating_code](#) ([she_hdl_t](#) session_hdl)

5.21.1 Detailed Description

We can get last rating code (error) using the below described function. This information is helpful during debugging an issue

5.21.2 Function Documentation

5.21.2.1 [she_get_last_rating_code\(\)](#)

```
uint32_t she_get_last_rating_code (  
    she\_hdl\_t session_hdl )
```

Report rating code from last command

SHE API defines standard errors that should be returned by API calls. Error code reported by SECO/V2X are "translated" to these SHE error codes. This API allow user to get the error code reported by SECO/V2X for the last command before its translation to SHE error codes. This should be used for debug purpose only.

Parameters

| | |
|--------------------|--------------------|
| <i>session_hdl</i> | SHE session handle |
|--------------------|--------------------|

Returns

rating code reported by last command

5.22 CMD_CANCEL

Functions

- void `she_cmd_cancel` (void)

5.22.1 Detailed Description

To cancel any running command, we can use below described command. This is helpful if we have mistakenly send an incorrect command and want to stop the execution of that command or discard the results if command has already been executed.

5.22.2 Function Documentation

5.22.2.1 `she_cmd_cancel()`

```
void she_cmd_cancel (  
    void )
```

interrupt any given function and discard all calculations and results.

5.23 Global Info

Macros

- `#define SOC_IMX8DXL 0xe`
- `#define SOC_IMX8ULP 0x84d`
- `#define SOC_IMX93 0x9300`
- `#define SOC_IMX95 0x9500`
- `#define SOC_REV_A0 0xa000`
- `#define SOC_REV_A1 0xa100`
- `#define SOC_REV_A2 0xa200`
- `#define SOC_REV_B0 0xb000`
- `#define SOC_LF_FAB_DEFAULT 0x1`
- `#define SOC_LF_FAB_MODE 0x2`
- `#define SOC_LF_NO_NXP_SECRETS 0x4`
- `#define SOC_LF_WITH_NXP_SECRETS 0x8`
- `#define SOC_LF_SCU_FW_CLOSED 0x10`
- `#define SOC_LF_SECO_FW_CLOSED 0x20`
- `#define SOC_LF_CLOSED 0x40`
- `#define SOC_LF_CLOSED_WITH_NXP_FW 0x80`
- `#define SOC_LF_PARTIAL_FIELD_RET 0x100`
- `#define SOC_LF_FIELD_RET 0x200`
- `#define SOC_LF_NO_RET 0x400`
- `#define GINFO_LIB_VERSION_LEN 16`
- `#define GINFO_NVM_VERSION_LEN 16`
- `#define GINFO_COMMIT_ID_SZ 40`
- `#define HSM_API_VERSION_1 0x1`
- `#define HSM_API_VERSION_2 0x2`

Functions

- void [populate_global_info](#) (uint32_t session_hdl)
- void [show_global_info](#) (void)
- bool [is_global_info_populated](#) (void)
- uint16_t [se_get_soc_id](#) (void)
- uint16_t [se_get_soc_rev](#) (void)
- uint16_t [se_get_chip_lifecycle](#) (void)
- uint8_t [se_get_fips_mode](#) (void)
- uint8_t [se_get_lib_newness_ver](#) (void)
- uint8_t [se_get_lib_major_ver](#) (void)
- uint8_t [se_get_lib_minor_ver](#) (void)
- uint8_t [se_get_nvm_newness_ver](#) (void)
- uint8_t [se_get_nvm_major_ver](#) (void)
- uint8_t [se_get_nvm_minor_ver](#) (void)
- const char * [se_get_commit_id](#) (void)
- const char * [se_get_lib_version](#) (void)
- const char * [se_get_nvm_version](#) (void)
- const char * [get_soc_id_str](#) (uint16_t soc_id)
- const char * [get_soc_rev_str](#) (uint16_t soc_rev)
- const char * [get_soc_lf_str](#) (uint16_t lifecycle)
- void [se_get_info](#) (uint32_t session_hdl, [op_get_info_args_t](#) *args)
- void [se_get_soc_info](#) (uint32_t session_hdl, uint16_t *soc_id, uint16_t *soc_rev)

5.23.1 Detailed Description

5.23.2 Function Documentation

5.23.2.1 populate_global_info()

```
void populate_global_info (
    uint32_t session_hdl )
```

Populate the Global Info structure

Parameters

| | |
|--------------------|--------------------------|
| <i>session_hdl</i> | identifying the session. |
|--------------------|--------------------------|

5.23.2.2 show_global_info()

```
void show_global_info (
    void )
```

Print the Global Info of library

5.23.2.3 is_global_info_populated()

```
bool is_global_info_populated (
    void )
```

Get the status of Global Info, if populated or not.

5.23.2.4 se_get_soc_id()

```
uint16_t se_get_soc_id (
    void )
```

Get SoC ID.

5.23.2.5 se_get_soc_rev()

```
uint16_t se_get_soc_rev (
    void )
```

Get SoC Revision.

5.23.2.6 se_get_chip_lifecycle()

```
uint16_t se_get_chip_lifecycle (  
    void )
```

Get Chip-lifecycle.

5.23.2.7 se_get_fips_mode()

```
uint8_t se_get_fips_mode (  
    void )
```

Get Fips mode.

5.23.2.8 se_get_lib_newness_ver()

```
uint8_t se_get_lib_newness_ver (  
    void )
```

Get library newness version.

5.23.2.9 se_get_lib_major_ver()

```
uint8_t se_get_lib_major_ver (  
    void )
```

Get library major version.

5.23.2.10 se_get_lib_minor_ver()

```
uint8_t se_get_lib_minor_ver (  
    void )
```

Get library minor version.

5.23.2.11 se_get_nvm_newness_ver()

```
uint8_t se_get_nvm_newness_ver (  
    void )
```

Get NVM newness version.

5.23.2.12 se_get_nvm_major_ver()

```
uint8_t se_get_nvm_major_ver (  
    void )
```

Get NVM major version.

5.23.2.13 se_get_nvm_minor_ver()

```
uint8_t se_get_nvm_minor_ver (
    void )
```

Get NVM minor version.

5.23.2.14 se_get_commit_id()

```
const char* se_get_commit_id (
    void )
```

Get Build commit id.

5.23.2.15 se_get_lib_version()

```
const char* se_get_lib_version (
    void )
```

Get library version string.

5.23.2.16 se_get_nvm_version()

```
const char* se_get_nvm_version (
    void )
```

Get NVM version string.

5.23.2.17 get_soc_id_str()

```
const char* get_soc_id_str (
    uint16_t soc_id )
```

Get the string representating SoC ID

Parameters

| | |
|---------------|---------------------------------|
| <i>soc_id</i> | SoC ID fetched from Global Info |
|---------------|---------------------------------|

Returns

String representation of the SoC ID

5.23.2.18 `get_soc_rev_str()`

```
const char* get_soc_rev_str (
    uint16_t soc_rev )
```

Get the string representating SoC Revision

Parameters

| | |
|----------------|---------------------------------------|
| <i>soc_rev</i> | SoC Revision fetched from Global Info |
|----------------|---------------------------------------|

Returns

String representation of the SoC Revision

5.23.2.19 `get_soc_lf_str()`

```
const char* get_soc_lf_str (
    uint16_t lifecycle )
```

Get the string representation of the Chip Lifecycle

Parameters

| | |
|------------------|--------------------------------|
| <i>lifecycle</i> | value fetched from Global Info |
|------------------|--------------------------------|

Returns

a string representation of Lifecycle

5.23.2.20 `se_get_info()`

```
void se_get_info (
    uint32_t session_hdl,
    op_get_info_args_t * args )
```

Get Info for Global Info setup

5.23.2.21 `se_get_soc_info()`

```
void se_get_soc_info (
    uint32_t session_hdl,
    uint16_t * soc_id,
    uint16_t * soc_rev )
```

Get SoC Info for Global Info setup

Index

- add_she_service
 - Session, [30](#)
- add_she_session
 - Session, [29](#)
- CMD_CANCEL, [55](#)
 - she_cmd_cancel, [55](#)
- CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB
 - / CMD_DEC_ECB, [9](#)
 - she_cipher_one_go, [11](#)
 - she_close_cipher_service, [11](#)
 - she_open_cipher_service, [11](#)
- CMD_EXPORT_RAM_KEY, [14](#)
 - she_export_plain_key, [14](#)
- CMD_EXTEND_SEED, [46](#)
 - she_extend_seed, [46](#)
- CMD_GENERATE_MAC, [16](#)
 - she_generate_fast_mac_mubuff_v2, [17](#)
 - she_generate_mac, [16](#)
- CMD_GET_ID, [20](#)
 - she_get_id, [20](#)
- CMD_GET_STATUS, [25](#)
 - she_get_status, [25](#)
- CMD_INIT_RNG, [42](#)
 - she_close_rng_service, [42](#)
 - she_open_rng_service, [42](#)
- CMD_LOAD_KEY, [37](#)
 - she_key_update, [38](#)
 - she_key_update_ext, [38](#)
- CMD_LOAD_PLAIN_KEY, [40](#)
 - she_load_plain_key, [40](#)
- CMD_RND, [44](#)
 - she_get_random, [45](#)
- CMD_VERIFY_MAC, [18](#)
 - she_verify_fast_mac_mubuff_v2, [19](#)
 - she_verify_mac, [19](#)
- delete_she_service
 - Session, [30](#)
- delete_she_session
 - Session, [29](#)
- Error codes, [49](#)
 - SHE_BUSY_RATING, [51](#)
 - SHE_CMD_NOT_SUPPORTED, [51](#)
 - she_err_t, [50](#)
 - SHE_FATAL_FAILURE, [51](#)
 - SHE_FEATURE_DISABLED, [51](#)
 - SHE_FEATURE_NOT_SUPPORTED, [51](#)
 - SHE_GENERAL_ERROR, [51](#)
 - SHE_ID_CONFLICT, [51](#)
 - SHE_INVALID_ADDRESS, [51](#)
 - SHE_INVALID_LIFECYCLE, [51](#)
 - SHE_INVALID_MESSAGE, [51](#)
 - SHE_INVALID_PARAM, [51](#)
 - SHE_KEY_EMPTY_RATING, [51](#)
 - SHE_KEY_INVALID_RATING, [51](#)
 - SHE_KEY_NOT_AVAILABLE_RATING, [51](#)
 - SHE_KEY_STORAGE_ERROR, [51](#)
 - SHE_KEY_STORE_AUTH, [51](#)
 - SHE_KEY_STORE_CONFLICT, [51](#)
 - SHE_KEY_STORE_COUNTER, [51](#)
 - SHE_KEY_UPDATE_ERROR_RATING, [51](#)
 - SHE_KEY_WRITE_PROTECTED_RATING, [51](#)
 - SHE_LIB_ERROR, [51](#)
 - SHE_MEMORY_FAILURE_RATING, [51](#)
 - SHE_NO_DEBUGGING_RATING, [51](#)
 - SHE_NO_ERROR, [51](#)
 - SHE_NO_SECURE_BOOT_RATING, [51](#)
 - SHE_NOT_READY, [51](#)
 - SHE_NVM_ERRO, [51](#)
 - SHE_OUT_OF_MEMORY, [51](#)
 - SHE_RNG_NOT_STARTED, [51](#)
 - SHE_RNG_SEED_RATING, [51](#)
 - SHE_SELF_TEST_FAILURE, [51](#)
 - SHE_SEQUENCE_ERROR_RATING, [51](#)
 - SHE_UNKNOWN_HANDLE, [51](#)
 - SHE_UNKNOWN_ID, [51](#)
 - SHE_UNKNOWN_KEY_STORE, [51](#)
 - SHE_UNKNOWN_WARNING, [51](#)
- get_soc_id_str
 - Global Info, [59](#)
- get_soc_lf_str
 - Global Info, [60](#)
- get_soc_rev_str
 - Global Info, [59](#)
- Global Info, [56](#)
 - get_soc_id_str, [59](#)
 - get_soc_lf_str, [60](#)
 - get_soc_rev_str, [59](#)
 - is_global_info_populated, [57](#)
 - populate_global_info, [57](#)
 - se_get_chip_lifecycle, [57](#)
 - se_get_commit_id, [59](#)
 - se_get_fips_mode, [58](#)
 - se_get_info, [60](#)
 - se_get_lib_major_ver, [58](#)
 - se_get_lib_minor_ver, [58](#)
 - se_get_lib_newness_ver, [58](#)

- se_get_lib_version, [59](#)
- se_get_nvm_major_ver, [58](#)
- se_get_nvm_minor_ver, [58](#)
- se_get_nvm_newness_ver, [58](#)
- se_get_nvm_version, [59](#)
- se_get_soc_id, [57](#)
- se_get_soc_info, [60](#)
- se_get_soc_rev, [57](#)
- show_global_info, [57](#)
- is_global_info_populated
 - Global Info, [57](#)
- Key store, [34](#)
 - she_close_key_store_service, [36](#)
 - she_open_key_store_service, [35](#)
- last rating code, [54](#)
 - she_get_last_rating_code, [54](#)
- op_cipher_one_go_args_t, [10](#)
- op_export_plain_key_args_t, [14](#)
- op_generate_mac_t, [16](#)
- op_get_id_args_t, [20](#)
- op_get_info_args_t, [22](#)
- op_get_random_args_t, [44](#)
- op_get_status_args_t, [25](#)
- op_key_store_reprov_en_args_t, [35](#)
- op_key_update_args_t, [37](#)
- op_key_update_ext_args_t, [37](#)
- op_load_plain_key_args_t, [40](#)
- op_open_utils_args_t, [52](#)
- op_rng_extend_seed_t, [46](#)
- op_shared_buf_args_t, [48](#)
- op_verify_mac_t, [18](#)
- open_session_args_t, [28](#)
- open_svc_cipher_args_t, [10](#)
- open_svc_key_store_args_t, [34](#)
- open_svc_rng_args_t, [44](#)
- populate_global_info
 - Global Info, [57](#)
- se_get_chip_lifecycle
 - Global Info, [57](#)
- se_get_commit_id
 - Global Info, [59](#)
- se_get_fips_mode
 - Global Info, [58](#)
- se_get_info
 - Global Info, [60](#)
- se_get_lib_major_ver
 - Global Info, [58](#)
- se_get_lib_minor_ver
 - Global Info, [58](#)
- se_get_lib_newness_ver
 - Global Info, [58](#)
- se_get_lib_version
 - Global Info, [59](#)
- se_get_nvm_major_ver
 - Global Info, [58](#)
- se_get_nvm_minor_ver
 - Global Info, [58](#)
- se_get_nvm_newness_ver
 - Global Info, [58](#)
- se_get_nvm_version
 - Global Info, [59](#)
- se_get_soc_id
 - Global Info, [57](#)
- se_get_soc_info
 - Global Info, [60](#)
- se_get_soc_rev
 - Global Info, [57](#)
- Session, [27](#)
 - add_she_service, [30](#)
 - add_she_session, [29](#)
 - delete_she_service, [30](#)
 - delete_she_session, [29](#)
 - she_close_session, [31](#)
 - she_hdl_t, [29](#)
 - she_open_session, [31](#)
 - she_service_hdl_to_ptr, [30](#)
 - she_session_hdl_to_ptr, [29](#)
- Shared Buffer, [48](#)
- SHE Commands, [24](#)
- SHE Get Info, [22](#)
 - she_get_info, [23](#)
- SHE keys, [32](#)
- SHE+ key extension, [33](#)
- SHE_BUSY_RATING
 - Error codes, [51](#)
- she_cipher_one_go
 - CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB, [11](#)
- she_close_cipher_service
 - CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB, [11](#)
- she_close_key_store_service
 - Key store, [36](#)
- she_close_rng_service
 - CMD_INIT_RNG, [42](#)
- she_close_session
 - Session, [31](#)
- she_close_utils
 - Utils, [53](#)
- she_cmd_cancel
 - CMD_CANCEL, [55](#)
- SHE_CMD_NOT_SUPPORTED
 - Error codes, [51](#)
- she_err_t
 - Error codes, [50](#)
- she_export_plain_key
 - CMD_EXPORT_RAM_KEY, [14](#)
- she_extend_seed
 - CMD_EXTEND_SEED, [46](#)
- SHE_FATAL_FAILURE
 - Error codes, [51](#)
- SHE_FEATURE_DISABLED

- Error codes, [51](#)
- SHE_FEATURE_NOT_SUPPORTED
 - Error codes, [51](#)
- SHE_GENERAL_ERROR
 - Error codes, [51](#)
- she_generate_fast_mac_mubuff_v2
 - CMD_GENERATE_MAC, [17](#)
- she_generate_mac
 - CMD_GENERATE_MAC, [16](#)
- she_get_id
 - CMD_GET_ID, [20](#)
- she_get_info
 - SHE Get Info, [23](#)
- she_get_last_rating_code
 - last rating code, [54](#)
- she_get_random
 - CMD_RND, [45](#)
- she_get_status
 - CMD_GET_STATUS, [25](#)
- she_hdl_t
 - Session, [29](#)
- SHE_ID_CONFLICT
 - Error codes, [51](#)
- SHE_INVALID_ADDRESS
 - Error codes, [51](#)
- SHE_INVALID_LIFECYCLE
 - Error codes, [51](#)
- SHE_INVALID_MESSAGE
 - Error codes, [51](#)
- SHE_INVALID_PARAM
 - Error codes, [51](#)
- SHE_KEY_EMPTY_RATING
 - Error codes, [51](#)
- SHE_KEY_INVALID_RATING
 - Error codes, [51](#)
- SHE_KEY_NOT_AVAILABLE_RATING
 - Error codes, [51](#)
- SHE_KEY_STORAGE_ERROR
 - Error codes, [51](#)
- SHE_KEY_STORE_AUTH
 - Error codes, [51](#)
- SHE_KEY_STORE_CONFLICT
 - Error codes, [51](#)
- SHE_KEY_STORE_COUNTER
 - Error codes, [51](#)
- she_key_update
 - CMD_LOAD_KEY, [38](#)
- SHE_KEY_UPDATE_ERROR_RATING
 - Error codes, [51](#)
- she_key_update_ext
 - CMD_LOAD_KEY, [38](#)
- SHE_KEY_WRITE_PROTECTED_RATING
 - Error codes, [51](#)
- SHE_LIB_ERROR
 - Error codes, [51](#)
- she_load_plain_key
 - CMD_LOAD_PLAIN_KEY, [40](#)
- SHE_MEMORY_FAILURE_RATING
 - Error codes, [51](#)
- SHE_NO_DEBUGGING_RATING
 - Error codes, [51](#)
- SHE_NO_ERROR
 - Error codes, [51](#)
- SHE_NO_SECURE_BOOT_RATING
 - Error codes, [51](#)
- SHE_NOT_READY
 - Error codes, [51](#)
- SHE_NVM_ERRO
 - Error codes, [51](#)
- she_open_cipher_service
 - CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB, [11](#)
- she_open_key_store_service
 - Key store, [35](#)
- she_open_rng_service
 - CMD_INIT_RNG, [42](#)
- she_open_session
 - Session, [31](#)
- she_open_utils
 - Utils, [52](#)
- SHE_OUT_OF_MEMORY
 - Error codes, [51](#)
- SHE_RNG_NOT_STARTED
 - Error codes, [51](#)
- SHE_RNG_SEED_RATING
 - Error codes, [51](#)
- SHE_SELF_TEST_FAILURE
 - Error codes, [51](#)
- SHE_SEQUENCE_ERROR_RATING
 - Error codes, [51](#)
- she_service_hdl_s, [28](#)
- she_service_hdl_to_ptr
 - Session, [30](#)
- she_session_hdl_s, [28](#)
- she_session_hdl_to_ptr
 - Session, [29](#)
- SHE_UNKNOWN_HANDLE
 - Error codes, [51](#)
- SHE_UNKNOWN_ID
 - Error codes, [51](#)
- SHE_UNKNOWN_KEY_STORE
 - Error codes, [51](#)
- SHE_UNKNOWN_WARNING
 - Error codes, [51](#)
- she_verify_fast_mac_mubuff_v2
 - CMD_VERIFY_MAC, [19](#)
- she_verify_mac
 - CMD_VERIFY_MAC, [19](#)
- show_global_info
 - Global Info, [57](#)
- Utils, [52](#)
 - she_close_utils, [53](#)
 - she_open_utils, [52](#)